

TREBALL FINAL DE GRAU



ESCOLA
POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Victor Gil Navarro

Titulació: Doble Grau en Enginyeria Informàtica i ADE

Títol de Treball Final de Grau: Optimizing a supply chain model of the pig sector and developing a decision support sytem

Director/a: Lluís Miquel Pla Aragones, Jordi Mateo Fornés

Presentació

Mes: Juny

Any: 2018

Victor Gil Navarro

Lluís Miquel Pla Aragues and Jordi Mateo Fornés

Dept. Maths and Dept. Computer Science

UdL (University of Lleida)

CONTENT

1. INTRODUCTION	4
2. STATE OF THE ART	6
3. METODOLOGY	7
3.1 MODEL.....	7
3.1.1 TECHNOLOGIES	7
3.1.2 MODEL DEFINITION.....	8
3.1.3 CPLEX OPTIONS CONFIGURATION.....	15
3.2 API AND DATABASE	21
3.2.1 TECHNOLOGIES	21
3.2.2 FUNCTIONALITIES.....	22
3.3 INTERFACE	25
3.3.1 TECHNOLOGIES	25
3.3.2 FUNCTIONALITIES.....	26
3.3.3 USE CASE STUDY.....	27
3.4 ARCHITECTURE	29
3.5 PROJECT MANAGEMENT	30
3.5.1 WORKFLOW.....	31
4. RESULTS AND ANALYSIS	33
4.1 PARAMETERS OF THE MODEL	33
4.2 INTERFACE AND API FUNCTIONALITIES.....	39
4.3 COMPUTATIONAL RESULTS.....	42
4.4 WORKFLOW	44
5. CONCLUSIONS	45
6. FUTURE WORK	46
7. BIBLIOGRAPHY.....	47
ANNEX I: OPTIONS FILE	49

Thanks to the Mathematics and the Computer Science Departments of the University of Lleida to allow me to take part in this project and guide me in the final stage of my double degree.

Special thanks to my project tutors Lluís Miquel Pla, Esteve Nadal, Jordi Mateo and Adela Pagès, who have helped and supported me in every possible way, solved any of my doubts and given me this opportunity.

ABSTRACT

This project consists in two parts due to the double degree in Computer Engineering and Administration and Business Management requirements. The first one presents and approaches a stochastic mixed integer linear programming model.

The model conceived for pig production planning has the objective to optimize the entire pig supply chain according to the number of farms operating for the same company or cooperative. The model maximizes the revenue calculated from the income of sales to the abattoir and the production costs. Production costs depends on each type of farm involved in the process. Factors like farm capacity, trucks available and transportation costs are considered, among others.

The model considers a medium-term planning horizon and specifically provides optimal transport planning in terms of number of animals to be transported from one facility to another and the number of trucks to do so.

The algebraic modelling software OPL studio, in combination with the solver CPLEX both from IBM ILOG have been used to solve the model and study its properties and behaviour. In addition, a potential improvement in the execution time has been sought considering that a real application of the model is expected.

This has been done through the generation of instances varying the number of farms that take part in each execution. The instances are required to apply a genetic algorithm on the search space defined by the cartesian product of the solver domains. Once a given time limit elapses, it returns the value of the parameters for which it has had a superior performance on average over the group of instances. Then, the results are analysed and the variations of the model execution time are studied.

The second part of the project consists in developing a decision support system (DSS), which takes form of an application that approaches and facilitates the use of the model in order to provide each user a personalized solution. In addition, a friendly interface displays a google maps personalized instance presenting all the facilities that users, companies and cooperatives or even farmers, have introduced to the system. The information introduced will be used to run the model and generate the specific solution for the user. In addition, the interface also allows users to introduce new facilities to the system and visualize their current information and characteristics.

1. INTRODUCTION

During the last years the pig sector has grown remarkably in production, population and number of farms, thanks to the demand of the foreign markets as well as the increase of the competitiveness of the sector worldwide. Spain has established itself as the second largest exporter of pigs in the European Union, drastically increasing the exports to third countries like China. In Spain, the pig sector represents the third part of the livestock sector while in Catalonia is even more important as it represents half of the livestock sector.

Companies in the sector and pig farmers depend on the income they get from the delivering of fattened pigs to the abattoir, so each decision made about the pigs from the sow farm to the abattoir may lead the company to succeed or to bankrupt. For this reason, optimization, simulation, decisions support systems (DSS) and big data technology are widely used to tackle these critical decisions where maximizing profit and reducing risk are the keystone for every business development plan.

This is where linear programming comes in play, as it has proven to be a vital tool for solving real-life problems in the recent years. A given problem is translated into a mathematical model where the main goal is to maximize or minimize a linear objective function within a set of linear constraints, which represent the practical and real bounds of the real-life problem. For instance, if one would develop a model to decide to invest either in 'a' or 'b' type funds depending on the refund, the variables would represent the amount invested in each type and the constraints would be the initial budget and the minimum amount to invest, among others.

Logistics, supply chain management, productions planning or portfolio optimization can benefit from linear programming. In the pig sector, in 2014, an article published in the American Society of Animal Science journal [10] described how a linear programming model was created to help a company take short-term decisions, like scheduling weekly transports of pigs between the farms, and mid and long-term decisions, like enlarging on shrinking the company or increasing or decreasing the farms capacity.

Large companies, whit tycoons supporting them, can afford to have a specialized team to create specific mathematical models to deal with their necessities and run them in powerful solvers and computers and make exhaustive analysis of the results. However, if the same necessity is transferred to SMEs (Small and Medium-sized Enterprises) or self-employed people who want to use these tools either for their activity or for personal reasons, the situation is somewhat more difficult to achieve, because despite of the fact that various free solvers are available, preparing and configuring a dedicated machine to run these solver executions is not possible for everyone, either because they lack the necessary economic or material resources or because the knowledge needed to configure, install and use these solvers is outwith their grasp.

Therefore, the combination of Cloud computing and Software as a Service (SAAS) and Linear Programming is an astonishing, powerful and synergic alliance (view Figure 1). The first two eliminate the necessity of a powerful computer with a dedicated environment to solve complex linear programming models. The only requirement is an electronic device with Internet connection. The user is able to obtain unlimited computing power, no matter which device is used, with a web browser.

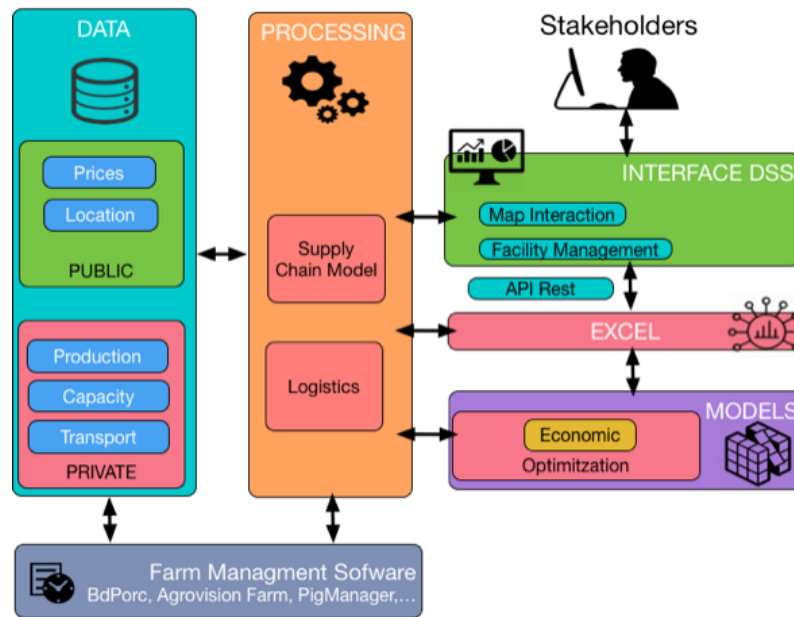


Figure 1: Software as a Service.

The application presented in this paper allows users, either companies or self-employed workers, to register their facilities introducing the coordinates, the capacity and the type of the facility saving the data in the Cloud. A friendly interface that makes use of Angular-Google-Maps technology allows users to visualize their facilities location and information in an interactive personalized map. This allows users to run the linear programming model personalized with their own facilities in the cloud. Moreover, it offers the possibility of customizing the machine where the solver will be executed. However, the service is limited to CPLEX OPL (Optimization Programming Language) compatible models. Therefore, the app takes form of a DSS to facilitate and approach the benefits of linear programming optimization to users.

As far as the execution is concerned, one would expect a short execution time. However, the execution of linear programming models can take a considerable amount of time to present a solution, long enough to be annoying for the user. For this reason, in this paper an exhaustive analysis is done to different execution instances in order to achieve an improvement in the execution timing, presented through specific configuration values of the CPLEX options file to run the model.

2. STATE OF THE ART

The development of decision support tools has been the subject of research for several decades. Already in the late 1950s descriptive and mathematical modelling was applied to solve agricultural problems [1]. Since then, researchers have developed new models aiming at improving any process chain, from the optimization of the feed supply [2] to the control of the farm climate [3] or the sow replacement problem [4], among others.

There are huge numbers of problems that can be solved or simplified with the help of optimization models. For instance, optimizing transportation networks to set vehicle routes to perform transportation at minimum cost is described in [5]. The authors in [6] propose a model capable of reducing costs and pollution in the iron industry (could be applied to the pig sector, analogously).

In addition, researchers continued to explore and develop advanced models with the objective of improving the overall performance of the pig supply chain [7, 8].

The integration of models adopting the whole vision of the production process is becoming more and more demanded. In this context, [9] proposed a first integer linear programming model to control the flow of animals in a three steps system structure but without taking into account transportation constraints. The model was reformulated in [10] to take into account transportation and the flow of the animals through different facilities. However, the new model neglected some important operational aspects for practical decision making: a marketing time window to deliver fattened pigs to the abattoir (according the sales price fluctuations); commercial value of carcasses by a grid reward system based on leanness, back-fat thickness and weight; batch management instead of continuous production; and finally, facility location and occupation.

The mathematical model with which this paper is based can be found in [10].

Current state-of-the-art commercial solvers for farm management tends to be adapted to the new times technology: multi-site, cloud-based services, etc (but with no relation with business intelligence). However, there are many reasons why a model does not end in farmers hands. As several authors claim, [11, 12], the main reasons can be summarized as: unclear focus of the project, lack of interface or hard system maintenance, long execution timings. This where DSS come to play to solve this issues.

From the boarder perspective of agricultural systems, Jones et al [13] who emphasizes the definition of a Use Case as the central pivot for the development of the interactions between models and users, and expose the difficulties for the final user to find easy accessible and usable applications.

Therefore, the main contribution of this project is presenting an app that allows to run a complex mathematical model, which englobes the whole supply chain management, and present the solution as a service to the farmers.

3. METODOLOGY

3.1 MODEL

3.1.1 TECHNOLOGIES

A far as concerns the model part, IBM ILOG CPLEX Optimization Studio version 12.7.1.0 has been used to run the model and generate the different instances for the later analysis. Other solvers can be found in the market, as XPRESS or GUBORI. However, the use of CPLEX was a requirement for the project.

Moreover, the use of VBA (Visual Basic for Applications) was required for the Microsoft Excel developing the automatization of the process.



Image 1: Model technologies. IBM ILOG CPLEX and VBA.

3.1.2 MODEL DEFINITION

The current companies of the pig sector are large integrators of the production system, which is called “in stages” (view Figure 2). This system separates all the production cycle in three types of facilities: the sow farm, the rearing farm and the fattening farm. The first one focuses on maximizing the birth of piglets. The rearing farm houses the weaned piglets and prepares them for the last phase. Lastly, the fattening farm, which as the name claims, has the objective that the piglets achieve marketable the target weight before the deliver to the abattoir.

This situation can correspond to a private company vertically integrated owning all or most of the farms or to a cooperative of associated producers owners of one or more farms.

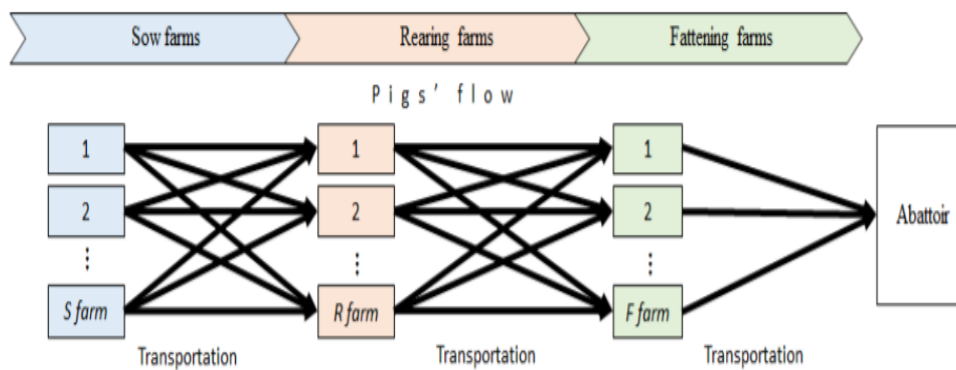


Figure 2: Pig production phases and pig flow.

This system allows each facility to specialize production, obtain benefits of an economy of scale and, also, take advantage of some health benefits because of the specialization and coordination of the agents within the chain. This helps to diversify the number of specialized farms, makes the integration easier in bigger companies, reduces risk of diseases, uncertainty and generates value. In addition, pigs are, usually, fattened in batches and all-in-all-out management strategy is common in commercial farms as this method maximizes the benefits specified.

The model has been extracted from “*Production planning of supply chains in pig industry*” by Esteve Nadal Roig, Lluís M. Pla and Antonio Alonso-Ayuso from the preprint submitted to Computers & Electronics in Agriculture.

Pig production is structured in three phases (maternity, rearing and fattening), encompassing different agents or farms.

At each phase, a set of specialized with their own characteristics and parameters, which include location and capacity among others, are considered. The first phase focuses on producing piglets in sow farms, the second focuses on rearing and weaning piglets in rearing farms and the third one focuses on fattening the piglets until they achieve the

target weight, getting them ready to deliver to the abattoir. Pigs remain for a certain number of weeks in order to ensure correct growth, with health and welfare conditions. Transfers between farms are performed by trucks, which are often contracted to a third party. The capacity of the trucks depends on the animals weight and is subjected to animal welfare EU regulations.

Weaned piglets of four weeks old and over seven kilograms of weight are transferred from sow farms to rearing farms, where they stay from four to eight weeks until they reach a weight of twenty to thirty kilos. Then they are transferred to fattening farms. In the last phase, after fifteen or eighteen weeks the piglets reach a marketable weight, around a hundred kilos. At that point, pigs are delivered to the abattoir.

The pig sector, currently operates under the batch management, which consists in assuring a number of farrows per week per sow by grouping sows in batches. Sows of the same batches are synchronized at the same reproductive stage. Batch management in fattening farms, also known as All-In-All-Out (AIAO), which allows the fattening farm to be filled and emptied at a time by a batch of pigs. In between batches, the facility is emptied and available for cleaning and disinfection. Batch management is one of the best modes to prevent the spread of diseases.

The model aims to maximize the revenue obtained in the PSC (Pig Supply Chain) by the farmers by selling the fattened pigs to the abattoir taking into account the costs of transporting the pigs between the farms and from fattening farms to the abattoir or the cost of maintaining the pigs in the different facilities.

First, the constants and decision variables will be defined. Then the model will be explained and finally the constraints will be presented.

Indexes and Sets

$t \in \mathcal{T}$, time horizon (in weeks), $t = 1, \dots, \mathcal{T}$.

$h \in \mathcal{H}$, , farms belonging the PSC, $h = 1, \dots, H$. $\mathcal{H} = \mathcal{H}_B \cup \mathcal{H}_R \cup \mathcal{H}_F$: Disjoint partition of farms in three phases (sites), being \mathcal{H}_B the set of sow farms, \mathcal{H}_R the set of rearing farms and \mathcal{H}_F the set of fattening farms.

$e \in \mathcal{E}$, Growing period in weeks, $e = 1, \dots, E$, where \mathcal{E} represents the productive cycle in weeks of a commercial pig from farrowing to the slaughtering. $\mathcal{E} = \mathcal{E}_B \cup \mathcal{E}_R \cup \mathcal{E}_F$: Disjoint partition of the productive cycle in each phase, i.e. weeks spent by pigs in different facilities, being $\mathcal{E}_B = \{1, \dots, E_B\}$ the set of weeks corresponding to the lactation period (from the birth to the weaning of piglets), $\mathcal{E}_R = \{E_B + 1, \dots, E_R\}$ set of weeks corresponding to the rearing period (from weaning to the beginning of the fattening) and $\mathcal{E}_F = \{E_R + 1, \dots, E_F\}$ set of weeks corresponding to the fattening period.

S : set of physiological states in which sow lifespan is divided.

$S_g \subset S$: set of farrowing states in the end of which farrowing take place and piglets born.

Parameters

$IN_{h,e}$: initial inventory of pigs of age e in the farm $h \in H$.

p^b_{ij} : transition probabilities of sows from i to j , with $i, j \in S$ in sow farm $b \in B$.

K_h : capacity in number of sows if $h \in B$ or pigs if $h \in R \cup F$.

LS_b : litter size at farrowing in the $b \in B$ sow farm.

C_p : unitary cost per week per piglet including feeding.

C_t : unitary cost per kilometre of a truck.

pp_t : expected pork value at week t .

ps_t : expected culled sow value at week t .

$d(h, h^*)$: distance from farm h to another farm or to the abattoir, $h^* \in H \cup \{a\}$.

ka_h : capacity in number of animals a truck can transport from farm h to the abattoir.

kg_h : capacity in number of animals a truck can transport from farm h to another farm.

$\pi_{b,i}$: steady state inventory of the total number of sows at physiological state $i \in S$ in the sow farm $b \in B$.

D_t : minimum weekly demand of the abattoir to be satisfied with fattened pigs.

Decision variables

$I_{t,e}$: total inventory of piglets of age $e \in E$ at week t in the system.

$I_{t,h,e}$: inventory of piglets of age $e \in E_h$ at week t on the farm h .

$A_{t,h}$: inventory of oldest pigs (greatest age) allowed on farm h , at week t to be transferred to the next stage in the chain.

$A_{t,b,r}$: inventory of piglets at week t transferred from $b \in B$ to $r \in R$ farm.

$A_{t,r,f}$: inventory of piglets at week t transferred from $r \in R$ to $f \in F$ farm.

$Nka_{t,h}$: number of trucks available at week t to transport animals leaving the farm h to the abattoir.

Nkg_{t,h_1,h_2} : number of trucks available at week t to transport piglets covering the path between farm h_1 and h_2 being $h_1 \in B$ and $h_2 \in R$ or $h_1 \in R$ and $h_2 \in F$.

Objective function

$$\begin{aligned} \max \sum_{h \in H} \sum_{t \in T} gm^h[t] = & \sum_{h \in H} \sum_{t \in T} (v^h[t] - c^h[t]) = \sum_{t \in T} \left(\sum_{b \in B} ps_t \pi_{ba} + \sum_{f \in F} pp_t A_{tf} \right) - \\ & \sum_{t \in T} \left(\sum_{h \in H} \sum_{e \in E} Cp \cdot I_{the} + \sum_{h \in H} Nka_{th} C_t d(h, a) + \sum_{b \in B} Nkg_{tb} C_t d(b, r) + \sum_{r \in R} Nkg_{tr} C_t d(r, f) \right) \end{aligned}$$

where:

$gm^h[t]$: Total weekly gross margin of farm h in the t week.

$v^h[t]$: Total weekly income of farm h in the t week, in particular sales to the abattoir of culled sows and fattened pigs.

$c^h[t]$: Total weekly cost of farm h in the t week, considering costs per sow and pigs and transport costs to the abattoir or between farms.

Constraints of the Model

Capacity of facilities

$$\sum_{i \in S} \pi_{b,i} \leq K_b \quad b \in B \quad (1)$$

$$\sum_{e \in E} I_{t,h,e} \leq K_h \quad t \in T, h \in H - B \quad (2)$$

Sow herd dynamics

$$\pi_{b,j} - \sum_{i \in S} p_{i,j} \pi_{b,i} = 0 \quad j \in S, b \in B \quad (3)$$

Initial Inventory

$$I_{0,h,e} = IN_{h,e} \quad e \in E, h \in H \quad (4)$$

Growth of animals

$$I_{t,e+1} = I_{t-1,e} \quad e \in E \setminus \{|E|\}, t \in T \setminus \{1\} \quad (5)$$

$$I_{t,b,e+1} = I_{t-1,b,e} \quad e \in E_B \setminus \{|E_B|\}, t \in T \setminus \{1\}, b \in B \quad (6)$$

$$I_{t,r,e+1} = I_{t-1,r,e} \quad e \in E_R \setminus \{|E_R|\}, t \in T \setminus \{1\}, r \in R \quad (7)$$

$$I_{t,f,e+1} = I_{t-1,f,e} \quad e \in E_F \setminus \{|E_F|\}, t \in T \setminus \{1\}, f \in F \quad (8)$$

Transfer between farms

$$A_{t,h} \leq I_{t,h,|E_h|} \quad t \in T, h \in H \quad (9)$$

$$I_{t,r,|E_R|+1} = \sum_{b \in B} A_{t-1,b,r} \quad t \in T \setminus \{1\}, r \in R \quad (10)$$

$$\sum_{r \in R} A_{t,b,r} = A_{t,b} \quad t \in T, b \in B \quad (11)$$

$$I_{t,f,|E_F|+1} = \sum_{r \in R} A_{t-1,r,f} \quad t \in T \setminus \{1\}, f \in F \quad (12)$$

$$\sum_{f \in F} A_{t,r,f} = A_{t,r} \quad t \in T, r \in R \quad (13)$$

Transportation capacity

$$\pi_{ba} \leq Nka_{t,b} * ka_b \quad t \in T, b \in B \quad (14)$$

$$A_{t,f} \leq Nka_{t,f} * ka_f \quad t \in T, f \in F \quad (15)$$

$$A_{t,h_1,h_2} \leq Nkg_{t,h_1,h_2} * kg_{h_1} \quad t \in T, h \in H - F \quad (16)$$

Litter size

$$I_{t,b,1} \leq \sum_{i \in S_g \subset S} \pi_{b,i} \cdot LS_{b,i} \quad t \in T, b \in B \quad (17)$$

Abattoir's quota

$$\sum_{f \in F} A_{t,f} \geq D_t \quad t \in T, f \in F \quad (18)$$

3.1.3 CPLEX OPTIONS CONFIGURATION

The IBM ILOG CPLEX Optimization Studio has been used to run and solve the model using the simplex method.

In order to run a linear programming problem, a file where the model is defined alongside with its constraints is required as well as a data file, which fetches the data with the corresponding information to the desired variable, in this case from an excel document. In addition, a configuration file may be added to specify the CPLEX Optimizer settings and adapt CPLEX values to the necessities. This file may become handy in many situations because the engines have default behaviours that are turned to perform well on average. However, this behaviour can be improved.

In this section, CPLEX parameters of the mentioned configuration that belong to the mathematical programming section will be explained. These are divided in the following subsections: General, Conflicts, Emphasis, FeasOpt, Preprocess, Read, Adjust, Simplex (the solver algorithm), Integer Programming, Barrier, Net and Sifting. Each section has its own attributes.

Mathematical Programming

General:

- Advanced start switch (CPLEX_PARAM_ADVIND): Specifies that CPLEX should use advanced starting when optimization is initiated.
- Type of solution (CPLEX_PARAM_SOLUTIONTYPE): Specifies the type of solution (basic or non basic) that CPLEX produces for a linear program.

Emphasis:

- MIP emphasis switch (CPLX_PARAM_MIPEMPHASIS): Controls trade-offs between speed, feasibility, optimality, and moving bounds in MIP.

FeasOpt:

- Mode of FeasOpt (CPX_PARAM_FEASOPTMODE): Decides how FeasOpt measures the relaxation when finding a minimal relaxation in an infeasible model.

Preprocess:

- Preprocessing aggregator application limit (CPX_PARAM_AGGIND): Invokes the aggregator to use substitution where possible to reduce the number of rows and columns before the problem is solved.

- Bound strengthening¹ switch (CPX_PARAM_BDNSTRENIND): Decides whether to apply bound strengthening in mixed integer programs (MIPs).
- Coefficient reduction setting (CPX_PARAM_COEREDIND): Decides how coefficient reduction is used.
- Dependency switch (CPX_PARAM_DEPIND): Decides whether to activate the dependency checker.
- Preprocessing aggregator fill (CPX_PARAM_AGGFILL): Limits variable substitutions by the aggregator.
- Primal and Dual reduction type (CPX_PARAM_REDUCE): Specifies whether primal, dual, both or neither reductions are performed during preprocessing. These are also known as presolve reductions.
- Relaxed LP presolve switch (CPX_PARAM_RELAXPREIND): Decides whether LP presolve is applied to the root relaxation in a mixed integer program (MIP).
- Symmetry breaking (CPX_PARAM_SYMMETRY): Decides whether symmetry breaking reductions will be automatically executed, during the preprocessing phase, in a MIP model. The higher, the more aggressive breaks.

Read

- Scale parameter (CPX_PARAM_SCAIND): Decides how to scale the problem matrix.

Simplex

- Dual simplex pricing algorithm (CPX_PARAM_DPRIND): Decides the type of pricing applied in the dual simplex algorithm.
- Primal simplex pricing algorithm (CPX_PARAM_PPRIIND): Sets the primal pricing algorithm.

¹ Strengthening means to replace one row of a model with another such that an integer vector is feasible in the new row if and only if the integer vector was feasible in the original row. This improves the relaxation of the row by finding a dominating row.

- Simplex refactoring frequency (CPX_PARAM_REINV): Sets the numbers of iterations between refactoring of the basis matrix.
- Perturbation constant (CPX_PARAM_EPPER): Sets the amount of by which CPLEX perturbs the upper and lower bounds or objective coefficients on the variables when a problem is perturbed in the simplex algorithm.
- Simplex singularity repair limit (CPX_PARAM_SINGLIM): Restricts the number of times CPLEX attempts to repair the basis when singularities are encountered during the simplex algorithm. When the limit is exceeded, CPLEX replaces the current basis with the best factorable basis that has been found.

Integer Programming

General

- MIP priority order generation (CPX_PARAM_MIPORDTYPE): Selects the type of generic priority order to generate when no priority order is present.
- MIP strategy best bound interval (CPX_PARAM_BBINTERVAL): Sets the best bound interval for MIP strategy. The best bound interval is the interval at which the best bound node, instead of the best estimated node, is selected from the tree.

Strategy

- MIP dive strategy (CPX_PARAM_DIVETYPE): Controls the MIP dive strategy. This strategy occasionally performs probing dives, where it looks ahead at both children nodes before deciding which node to choose.
- Feasibility pump switch (CPX_PARAM_FPHEUR): Turns on or of the feasibility pump heuristic² for MIP models.
- MIP heuristic frequency (CPX_PARAM_HEURFREQ): Decides how often to apply the periodic heuristic.

² For more detail about the feasibility pump heuristic, see research by Fischetti, Glover, and Lodi (2003, 2005), by Bertacco, Fischetti, and Lodi (2005), and by Achterberg and Berthold (2005, 2007).

- Node presolve switch (CPX_PARAM_PRESLVND): Decides whether node presolve should be performed at the nodes of a MIP solution.
- MIP probing level (CPX_PARAM_PROBE): Sets the amount of probing on variables to be performed before MIP branching. This can be very powerful but very time-consuming at the start leading to dramatic reductions or dramatic increases in solution time.
- RINS heuristic frequency (CPX_PARAM_RINSHEUR): Decides how often to apply the relaxation induced neighbourhood search (RINS) heuristic. This heuristic attempts to improve upon the best solution found so far. It will not be applied until CPLEX finds one incumbent solution.
- MIP starting algorithm (CPX_PARAM_STARTALG): Sets which continuous optimizer will be used to solve the initial relaxation of a MIP.
- MIP subproblem algorithm (CPX_PARAM_SUBALG): Decides which continuous optimizer will be used to solve the subproblems in a MIP, after the initial relaxation
- MIP variable selection strategy (CPX_PARAM_VARSEL): Sets the rule for selecting the branching variable at the node which has been selected for branching.
- MIP dynamic search switch (CPX_PARAM_MIPSEARCH): Sets the search strategy for a mixed integer program.

Limits

- Constraint aggregation limit for cut generation (CPX_PARAM_AGGCUTLIM): Limits the number of constraints that can be aggregated for generating flow cover and mixed integer rounding (MIR) cuts.
- Candidate limit for generating Gomory fractional cuts (CPX_PARAM_FRACCAND): Limits the number of candidate variables for generating Gomory fractional cuts.
- MIP strong branching candidate list limit (CPX_PARAM_STRONGCANDLIM): Controls the length of the

candidate list when CPLEX uses variable selection as the setting for strong branching.

- Limit on nodes explored when CPLEX solves a subMIP (CPX_PARAM_SUBMIPNODELIMIT): Limits the number of nodes explored when CPLEX solves a subMIP.

Cuts

- MIP cliques switch: (CPX_PARAM_CLIQUES): Decides whether clique³ cuts should be generated for the problem.
- MIP covers switch: (CPX_PARAM_COVERS): Decides whether cover cuts⁴ should be generated for the problem.
- MIP disjunctive cuts switch (CPX_PARAM_DISJCUTS): Decides whether disjunctive cuts⁵ should be generated for the problem.
- MIP flow cover cuts switch (CPX_PARAM_FLOWCOVERS): Decides whether disjunctive cuts⁶ should be generated for the problem.
- MIP Gomory fractional cuts switch (CPX_PARAM_FRACCUTS): Decides whether Gomory fractional cuts⁷ should be generated for the problem.

³ A clique is a relationship among a group of binary variables such that at most one variable in the group can be positive in any integer feasible solution. Before optimization starts, CPLEX constructs a graph representing these relationships and finds maximal cliques in the graph.

⁴ If a constraint takes the form of a knapsack *constraint* (that is, a sum of binary variables with nonnegative coefficients less than or equal to a nonnegative righthand side), then there is a minimal cover associated with the constraint. A *minimal cover* is a subset of the variables of the inequality such that if all the subset variables were set to one, the knapsack constraint would be violated, but if any one subset variable were excluded, the constraint would be satisfied. CPLEX can generate a constraint corresponding to this condition, and this cut is called a cover cut.

⁵ A MIP problem can be divided into two subproblems with disjunctive feasible regions of their LP relaxations by branching on an integer variable. *Disjunctive cuts* are inequalities valid for the feasible regions of LP relaxations of the subproblems, but not valid for the feasible region of LP relaxation of the MIP problem.

⁶ *Flow covers* are generated from constraints that contain continuous variables, where the continuous variables have variable upper bounds that are zero or positive depending on the setting of associated binary variables.

⁷ *Gomory fractional cuts* are generated by applying integer rounding on a pivot row in the optimal LP tableau for a (basic) integer variable with a fractional solution value.

- MIP implied bound cuts switch (CPX_PARAM_IMPLBD): Decides whether implied bound cuts⁸ should be generated for the problem.
- MIP mixed integer rounding (MIR) switch (CPX_PARAM_MIRCUTS): Decides whether MIR cuts⁹ should be generated for the problem.
- MCF cut switch (CPX_PARAM_MCFCUTS): Switches on or off the generation of multi-commodity flow cuts¹⁰ in a MIP.
- MIP flow path cut switch (CPX_PARAM_FLOWPATHS): Decides whether flow path cuts¹¹ should be generated for the problem.

Grouping of Solutions

- Solution pool intensity (CPX_PARAM_SOLNPOOLINTENSITY): Controls the trade-off between the number of solutions generated for the solution pool and the amount of time or memory consumed. This parameter applies both to MIP optimization and to the populate procedure.

Barrier

- Barrier algorithm (CPX_PARAM_BARALG): Standard Barrier, which is the fastest algorithm for the model.
- Barrier crossover algorithm (CPX_PARAM_BARCROSSALG): Decides which crossover is performed at the end of a barrier optimization. This parameter is used when CPLEX uses Barrier Optimizer to solve an LP or an QP problem.
- Barrier ordering algorithm (CPX_PARAM_BARORDER): Sets the algorithm to be used to permute the rows of the constraint matrix in order to reduce fill in the Cholesky factor.

⁸ In some models, binary variables imply bounds on continuous variables. CPLEX generates potential cuts to reflect these relationships.

⁹ *MIR cuts* are generated by applying integer rounding on the coefficients of integer variables and the righthand side of a constraint.

¹⁰ When a network structure is present, CPLEX generates cuts that state that the capacities installed on arcs pointing into a component of the network must be at least as large as the total flow demand of the component that cannot be satisfied by flow sources within the component.

¹¹ *Flow path cuts* are generated by considering a set of constraints containing the continuous variables that define a path structure in a network, where the constraints are nodes and the continuous variables are in-flows and out-flows. The flows will be on or off depending on the settings of the associated binary variables.

3.2 API AND DATABASE

3.2.1 TECHNOLOGIES

PyCharm, developed by JetBrains, is the IDE (Integrated Development Environment) used in this project for the core-api. It is a specific IDE for Python released under the Apache License.

In addition, Postman has been used as a support tool when building the RESTful API and for testing afterwards. It offers a sleek user interface with which to make HTML requests, without the hassle of writing a bunch of code just to test an API's functionality.

Finally, the core application and server side has been developed in Python and the DataBase in MySQL supported by SQLAlchemy. Also Falcon has been used as the framework for the backend, which has facilitated the building of the api. Falcon is a bare-metal Python web API framework that encourages the REST architectural style as resource classes implement HTTP method handlers that resolve requests and perform state transitions, which has resulted very handy for the project.



Image 2: Api and Database technologies. PyCharm, Falcon, Postman and SqlAlchemy, respectively.

3.2.2 FUNCTIONALITIES

Backend Protection

In order to protect the user credentials, in the client, the user password is encoded using a hash in base 64. Authorization header is used with value “Basic user:password” where the user and the password are encoded. So the password remains encoded in the backend. Therefore, if the backend would be attacked and the password retrieved by any malicious ways, only the hash would be obtained.

To add another layer of security, in the backend a token is generated to every user that logs in, which is used to perform any of the functionalities that require a logged user. The token acts like an electronic key to access something, which is used in addition of the password.

To make any petition to the api that requires authorization, the token will be required as the value of the Authorization header of the petition. In an attempt to reduce any potential damage done to a user, the token expires when the user session is ended or after 24 hours since its generation.

Database Schema

The database schema of the project can be found in Figure 3.

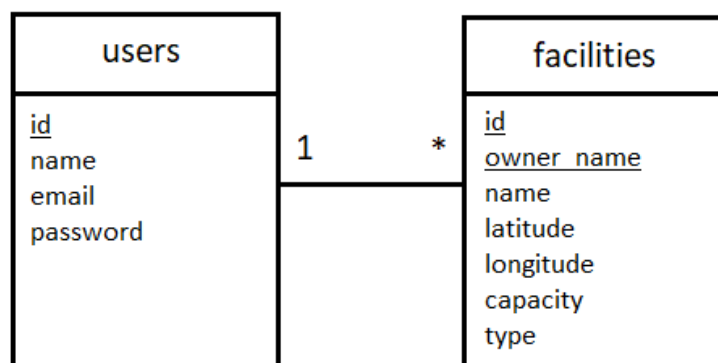


Figure 3: Database schema.

REST petitions

The api responds to the REST petitions found in Table 1. The petitions that require the Authentication header must have the user token as value.

Route	Type	Parameters	Auth Required	Response
/facilities/search	GET	-	YES	List of the user facilities
/facilities/show/{id:int}	GET	id:number, owner_name:string	YES	Facility
/facilities/register	PUT	Body: owner_name:string, name:string, latitude:number, longitude:number, type:string	YES	-
/user/login	POST	Header: Authorization: Basic Auth user:password	NO	{ status, message, token }
/user/register	POST	Body: name:string, password:string, email:string	NO	-

Table 1: REST petitions.

API

You can invoke server code deployed in the cloud via the client app. The main functionalities of the api server are the following ones.

Register new users

Given a name, a password and an email, the server registers a new user with this information. The password is encoded using a hash in base 64 in the client. The server decodes it.

Login

When a registered user logs in throw the client app, the server generates a unique token for the user, which is required to perform any of the functionalities

that require to be logged, like adding a new facility, for instance. This token expires when the user session is closed. Otherwise, it expires after 24 hours.

Register new facilities

New facilities are saved in the data base with an additional parameter, the user that registered the mentioned facility.

Get user facilities

In order to display the facilities on the map, the api provides all the facilities and their information of a concrete user.

3.3 INTERFACE

3.3.1 TECHNOLOGIES

The chosen IDE for the frontend development has been WebStorm, also developed by JetBrains. WebStorm also provides code analysis, a graphical debugger, an integrated unit tester, integration with VCS, Version Control Systems, (Github is the one used for this project) and supports web development.

Angular 4 has been used to develop the client applications. It is a TypeScript based open source front-ended web application platform led by the Angular Team at Google and by a community of individuals and corporations. Angular 4 final version was released on March 2017, it is compatible backwards with Angular 2. Among others, it introduces the HttpClient module, which is easier and more powerful library for making HTTP requests. Angular Google Maps, which is a set of directives (part of angular-ui), written in CoffeeScript and JavaScript which integrate Google Maps in AngularJS applications. It is based on the Google Maps JavaScript API version 3. Some of the features that includes are directives for most of the widely-used Google Maps objects, including markers, windows, lines and shapes.



Image 3: Interface technologies. WebStorm, Angular4 and Google Maps, respectively.

3.3.2 FUNCTIONALITIES

Client application

In this section, the main functionalities of the client app will be explained.

Login service

In essence, as many other login services, makes a petition to the api with the credentials entered. If the user and password are correct, the user logs in. The login service makes use of Authorization headers, in concrete Basic Auth.

Register service

A new user can be registered providing a name, an email and a password. If the user name does alr, the new user register will not be successful. The client is, also, the responsible of encoding the password in hash base 64.

Add facility

The user must be logged in to perform a register of a new facility. Moreover, the name of the facility, the capacity, the type of facility (SF, RF, FF, A, FP) and the location (latitude and longitude) must be provided.

Map interaction

In the map, all the facilities that a user has registered are displayed. Depending on the type of the facility, a different icon will represent it.

In addition, by clicking on the facility icon, all the information related with it will be displayed in form of a marker info-window.

In addition, when adding a new facility, the map can be clicked in order to introduce a temporal marker in the specific location. The marker can be dragged and moved around to the desired position. The marker information can be changed via the form above the map and when clicking the update info button, all the changes applied will be updated.

Get user token

The token is also provided to the user via a pop-up to enable him to run the excel macro, where the facilities information is fetched and then, the data file required to run the model is generated.

3.3.3 USE CASE STUDY

The following section presents a use case that captures how a user would interact with the solution to achieve a specific goal step by step, in this case, registering a new facility.

Use case analysis is an important and valuable requirement analysis technique that has an inherent, incremental and evolutionary nature, which fits well for agile development.

Use Case Name: Register Facility

Actors:

Registered User: Has an existing account with possibly other facilities registered.

Facility system: Processes petitions related to facilities.

Auth system: Checks if the user credentials are correct and manages the user's information and its token.

Triggers:

The user indicates that desires to register the facility.

Preconditions:

The user is logged in.

Post-conditions:

The facility will be registered in the system.

The new facility will be displayed in the map with the rest of the user's facilities.

Normal Flow:

1. The user clicks the add facility button.
2. The system shows a form to introduce the new facility information.
3. The system enables the user to mark the facility location in the map.
4. The user introduces the facility information.
5. The system checks the introduced data is correct and logic.
6. The user confirms registering the facility.
7. The system will submit the petition to the register system.
8. The system indicates the user that the facility has been registered.
9. The user exits the system.

Alternate Flows:

4A1: The user realizes an error in the facility information and will change it.

1. The user selects the form with the wrong information.
2. The user rewrites the right information.

3. The new information is displayed in the map.
4. The use case returns to step 5.

5A1: The user determines that the facility is not ready to be registered and will cancel the process.

1. The user clicks the cancel button.
2. The system hides the form and disables the map interaction.
3. The user case ends.

Exception Flows:

6A1: The user tries to register a facility with wrong information format.

1. The system detects the wrong information.
2. The system alerts the user to change the information.
3. The use case returns to step 4.

3.4 ARCHITECTURE

This section presents the architecture in which the client application has been developed. First, the architecture is analysed, and then the different parts that make up the cloud service are discussed. These include the Presentation Layer (user), the Business Logic and the Data Layer (view Figure 4).

The platform chosen to develop this architecture was OpenNebula. The main reason of this choice was to take advantage of the Stormy server due to is a private cloud deployed with the OpenNebula platform that belongs to the University of Lleida. However, what is most relevant is that Stormy is free of cost.

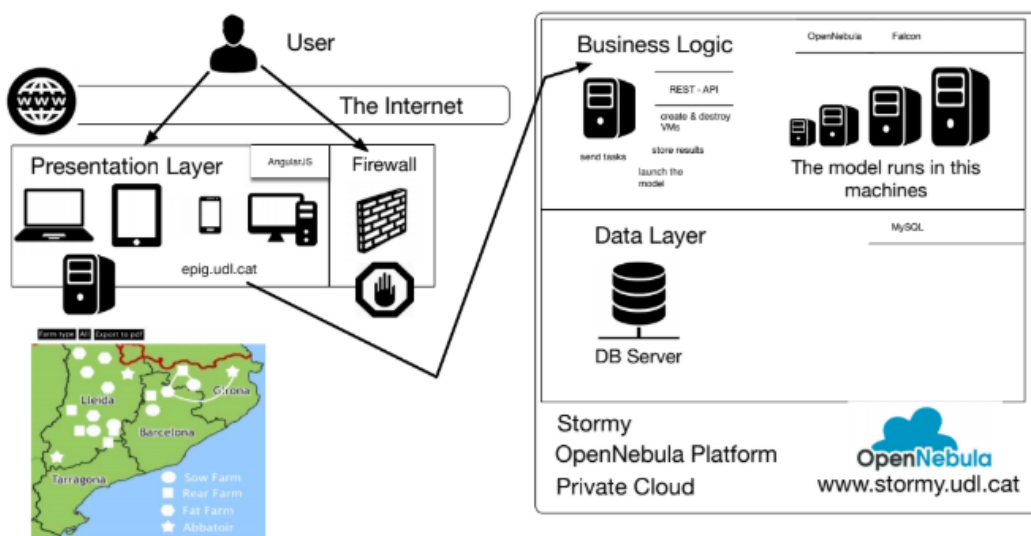


Figure 4: Project architecture.

The presentation layer provides the application's user interface (UI). This layer displays information related to such services as the facilities that the user has registered. It communicates with other business and data layers by which it puts out the results to the user interface. In other words, it displays and receives data from the user.

The business logic layer implements the functionality of the application by performing detailed processing. It also orchestrates and coordinates the data flow between the presentation and the data layers.

The data layer includes the persistence mechanisms like database servers and the data access layer that encapsulates the persistence mechanisms and exposes the data. The data access layer should provide an API to the application that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms. Avoiding dependencies facilitates future updates or edits and even makes other layers not being aware of these changes.

3.5 PROJECT MANAGEMENT

As the development of this project involved several areas, required teamwork between different departments and the project need's may change over time, it was decided to use SCRUM as an agile methodology.

Agile methodologies make reference to software engineering methods based on iterative and incremental development, where requirements and solutions evolve over time. The work is carried out through the collaboration of a multidisciplinary team, immersed in a shared process of short-term decision-making. The development of the project is divided in iterations, called sprints, which include planning, requirement analysis, design, coding, testing and documentation. In addition, the different phases of development are overlapped, instead of performing one after other in a sequential or cascade cycle. Another aspect of agile methodologies is that face-to-face communication is emphasized instead of documentation.

SCRUM offers a series of benefits among which stand out a better quality as the continual feedback and exposure ensure that it is as high as possible, faster solution delivery (about a 40% time save) and consequently increased return on investment because revenue start coming in sooner, increased collaboration and project performance due to the interaction between the teams that take part in the project development, increased project control and risk reduction.

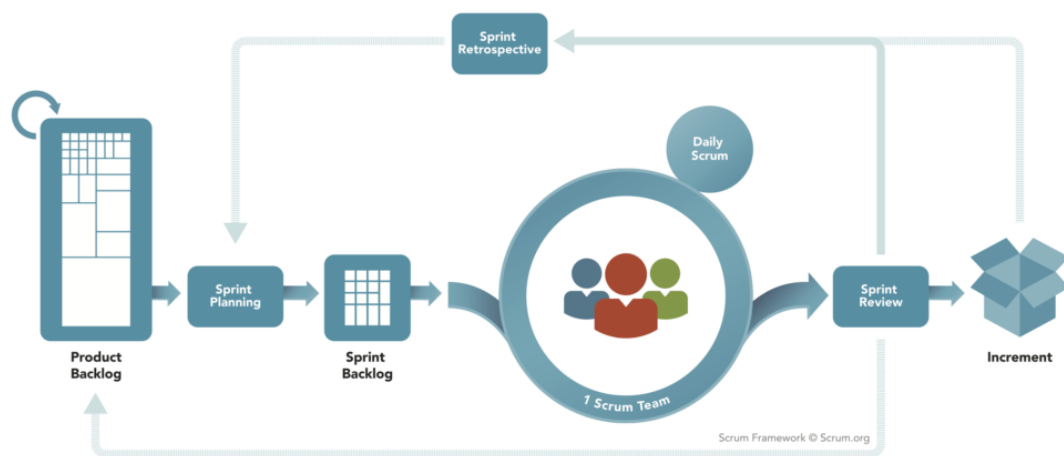


Figure 5: SCRUM based project development.

3.5.1 WORKFLOW

The initial work distribution of this project has been divided in the following tasks and steps. The tasks make reference to the client application and the api development and the steps to the work related with the model optimization.

Task 1: Prepare an Angular 4 App. [Angular, JavaScript, CSS and HTML]

1. Create a new project.
2. Be familiar with Angular.

Task 2: Map. [Angular, JavaScript, CSS and HTML]

1. Be familiar with shapefiles, maps and coordinates.
2. Plot the map.
3. Draw all the items on the map.
4. Put a legend.
5. Add item information.

Task 3: Use epig-core API. [Python]

1. Be familiar with the epig-core API.
2. Update the API.
3. Expand the DB.
4. Integrate epig-core API to obtain the information.

Task 4: Add Login/Register functionality. [Angular, Javascript, CSS and HTML]

Task 5: Link the api to the Angular App. [Python, Angular]

Task 6: Develop Excel macros to automatize the process. [Visual Basic for Applications]

1. Macro that fetches all the user data from the api given a user name and a token.
2. Macro that generates the data file to run the model.

Task 7: Apply CSS to the application. [Angular, CSS and HTML]

Step 1: Prepare the environment to work with the model. [CPLEX]

1. Be familiar with CPLEX and linear programming.
2. Understand the model and its constraints.

Step 2: Apply changes to the model and study the response.

Step 3: Generate the instances to study the model behaviour and proceed to the analysis.

Step 4: Apply the new configuration and study the new behaviour.

Grant Diagram

In the following diagram the initial tasks and steps time distribution are displayed.

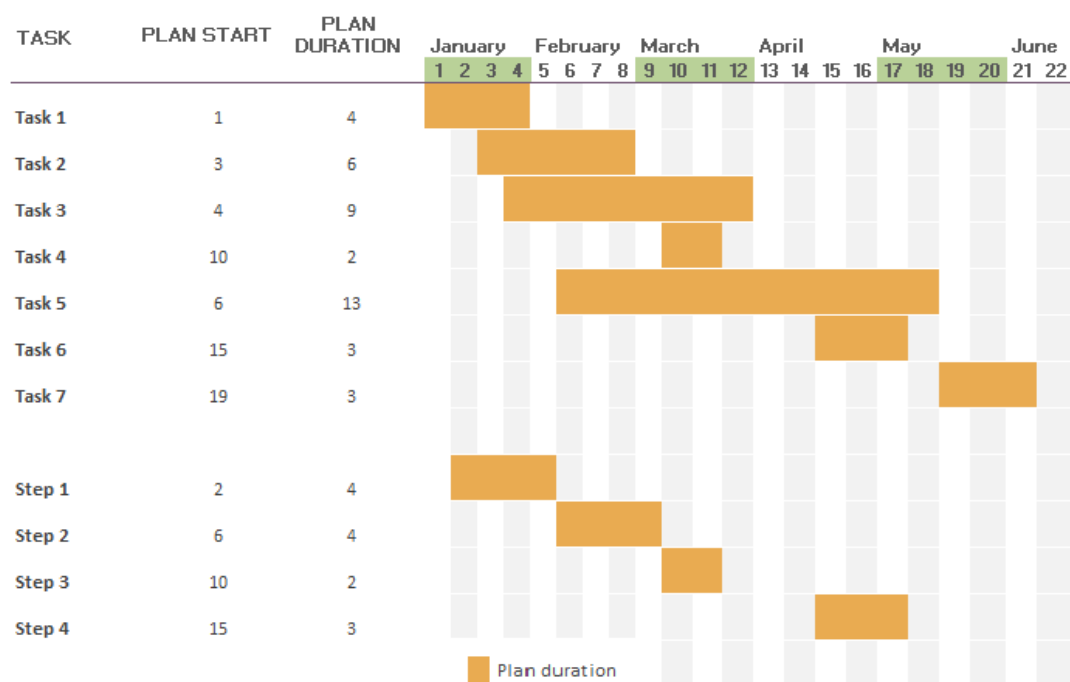


Figure 6: Grant Diagram.

4. RESULTS AND ANALYSIS

4.1 PARAMETERS OF THE MODEL

Seeking a real application of the model, a short execution time is expected. In an attempt to reduce the execution time of the model twenty samples, each one with different execution time and different number of facilities of each type, have been created in order to provide a wide range of possible executions with different results.

Instance	Sow Farms	Rearing Farms	Fattening Farms	Execution time (s)
1	2	6	12	5
2	2	6	20	9
3	2	8	20	13
4	2	10	40	14
5	2	15	80	29
6	3	8	15	31
7	3	6	10	39
8	3	15	20	45
9	4	12	20	55
10	3	15	40	71
11	3	15	90	82
12	3	15	55	97
13	3	15	60	107
14	3	18	80	137
15	4	12	40	159
16	4	15	60	173
17	3	21	82	183
18	4	15	40	200
19	4	8	15	203
20	3	21	85	216
21	4	18	85	228
22	3	21	80	234
23	4	12	80	247
24	4	18	60	256
25	5	18	50	301
26	5	18	60	304
27	5	21	70	473
28	5	18	70	496
29	5	18	90	564
30	5	18	80	628

Table 2: Instances execution time.

The execution time of the instances generated (view Table 2) oscillates between thirty seconds and three hundred seconds.

The fact that no other variables change apart from the number of facilities of each type is vital for the success of the later study.

Before running the analysis, the configuration parameter relative MIP gap tolerance was set to 0.05. This value sets a relative tolerance between the best integer objective and the objective of the best node remaining. When the value

$$\frac{|bestBound - bestInteger|}{1^{-10} + |bestInteger|}$$

falls below the value of this parameter, the mixed integer optimization is stopped. Therefore, the obtained solution will not be optimal but a close approximation. For the solver, the closer to the optimum the harder to keep advancing towards it, so with this parameter setting the solver task is soften.

With this precondition, the data study is started. In essence, this analysis consists in applying a genetic algorithm on the search space defined by the cartesian product of the domains of the solver parameters. Exhausted a limit time, returns the value of the parameters for which it has found better average performance on the set of test instances.

The winning configuration for the given attribute MIP gap tolerance equal to 0.05 has been the following one. Take in consideration that the configuration parameters that remain equal to the default ones are not mentioned. Even though, in the “Annex I” the whole configuration parameters values can be found.

Mathematical Programming:

General:

- Advanced start switch (CPLEX_PARAM_ADVIND = 0): No advanced start.
- Type of solution (CPLEX_PARAM_SOLUTIONTYPE = 2): Non basic solution. With this option, CPLEX seeks a pair of primal-dual solution vectors and CPLXE possibly may not produce basic status information (does not prevent it) about which variables and constraints participate in the basis.

Emphasis:

- MIP emphasis switch (CPLX_PARAM_MIPEMPHASIS = 4): Emphasise finding hidden feasible solutions. With this mode, the MIP optimizer works hard to find high quality feasible solutions that are otherwise very difficult to find.

FeasOpt:

- Mode of FeasOpt (CPX_PARAM_FEASOPTMODE = 4): Minimize the sum of squares of required relaxations in first phase and execute second phase to find optimum among minimal relaxations.

Preprocess:

- Preprocessing aggregator application limit (CPX_PARAM_AGGIND = 6): Apply aggregator 6 times.
- Bound strengthening switch (CPX_PARAM_BDNSTRENIND = 1): Apply bound strengthening.
- Coefficient reduction setting (CPX_PARAM_COEREDIND = 2): Reduce all potential coefficients. It reduces the coefficients, somewhat more aggressively, reducing every possible coefficient.
- Dependency switch (CPX_PARAM_DEPIND = 0): Turn off dependency checker. This prevents the dependency checker to search for dependent rows during preprocessing.
- Preprocessing aggregator fill (CPX_PARAM_AGGFILL = 785): At maximum, 785 substitutions to be made.
- Primal and Dual reduction type (CPX_PARAM_REDUCE = 0): No primal or dual reductions are done during preprocessing.
- Relaxed LP presolve switch (CPX_PARAM_RELAXPREIND = 1): Active. Uses presolve on initial relaxation.
- Symmetry breaking (CPX_PARAM_SYMMETRY = 0): Symmetry breaking deactivated.

Read

- Scale parameter (CPX_PARAM_SCAIND = -1): No scaling.

Simplex

- Dual simplex pricing algorithm (CPX_PARAM_DPRIND = 4): Steepest-edge pricing, unit initial norms.
- Primal simplex pricing algorithm (CPX_PARAM_PPRIIND = -1): Reduced cost-pricing.
- Simplex refactoring frequency (CPX_PARAM_REINV = 409): The number of iterations between refactoring of the basis matrix is 409.
- Perturbation constant (CPX_PARAM_EPPER = 5): 5 is the amount of by which CPLEX perturbs the upper and lower bounds or objective coefficients on the variables when a problem is perturbed in the simplex algorithm.
- Simplex singularity repair limit (CPX_PARAM_SINGLIM = 167):

Integer Programming

General

- MIP priority order generation (CPX_PARAM_MIPORDTYPE = 2): The priority order type chosen is by the use increasing bound range.
- MIP strategy best bound interval (CPX_PARAM_BBINTERVAL = 944): Selects the best bound node least frequently than best estimated node. This high value means that the best bound node will be selected less frequently because experience has shown it to be beneficial to select it occasionally.

Strategy

- MIP dive strategy (CPX_PARAM_DIVETYPE = 3): Guided dive. It spends more time exploring potential solutions that are similar to the current incumbent.
- Feasibility pump switch (CPX_PARAM_FPHEUR = -1): Turns off the feasibility pump heuristic.
- MIP heuristic frequency (CPX_PARAM_HEURFREQ = 3575): Apply the periodic heuristic with a frequency of 3575 (node interval), meaning that the heuristic will be called every 3575 nodes.

- Node presolve switch (CPX_PARAM_PRESLVND = -1): No node presolve. This reduces runtime considerably.
- MIP probing level (CPX_PARAM_PROBE = -1): No probing.
- RINS heuristic frequency (CPX_PARAM_RINSHEUR = 6969): Apply the periodic heuristic with a frequency of 6969 (node interval).
- MIP starting algorithm (CPX_PARAM_STARTALG = 4): The MIP starting algorithm chosen is Barrier.
- MIP subproblem algorithm (CPX_PARAM_SUBALG = 4): As well as for the MIP starting algorithm the algorithm chosen is Barrier.
- MIP variable selection strategy (CPX_PARAM_VARSEL = 1): Branch on variable with maximum infeasibility. The maximum infeasibility chooses the variable with the value furthest from an integer, which forces larger changes earlier in the tree.
- MIP dynamic search switch (CPX_PARAM_MIPSEARCH = 2): Apply dynamic search.

Limits

- Constraint aggregation limit for cut generation (CPX_PARAM_AGGCUTLIM = 347): The constraints limit to be aggregated is 347.
- Candidate limit for generating Gomory fractional cuts (CPX_PARAM_FRACCAND = 2906): The limit candidate variables for Gomory cuts is 2906.
- MIP strong branching candidate list limit (CPX_PARAM_STRONGCANDLIM = 40): The maximum length of the candidate list when CPLEX uses variable selection as the setting for strong branching is 40.
- Limit on nodes explored when CPLEX solves a subMIP (CPX_PARAM_SUBMIPNODELIMIT = 2038): The node limit when solving a subMIP is 2034.

Cuts

- MIP cliques switch: (CPX_PARAM_CLIQUES = 1): Generate clique cuts moderately.
- MIP covers switch: (CPX_PARAM_COVERS = -1): No cover cuts are generated.
- MIP disjunctive cuts switch (CPX_PARAM_DISJCUTS = 2): Generate disjunctive cuts with aggressivity.
- MIP flow cover cuts switch (CPX_PARAM_FLOWCOVERS = 2): Generate flow cover cuts with aggressivity.
- MIP Gomory fractional cuts switch (CPX_PARAM_FRACCUTS = 2): Generate Gomory fractional cuts with aggressivity.
- MIP implied bound cuts switch (CPX_PARAM_IMPLBD = 2): Generate implied bound cuts with aggressivity.
- MIP mixed integer rounding switch (CPX_PARAM_MIRCUTS = -1): No mixed integer rounding (MIR) cuts are generated.
- MCF cut switch (CPX_PARAM_MCFCUTS = 2): Generate multi-commodity flow (MCF) cuts with aggressivity.
- MIP flow path cut switch (CPX_PARAM_FLOWPATHS = 1): Generate implied bound cuts with moderation.

Grouping of Solutions

- Solution pool intensity (CPX_PARAM_SOLNPOOLINTENSITY = 4): Very aggressive intensity. Generates all solutions to the model.

Barrier

- Barrier algorithm (CPX_PARAM_BARALG = 2): Infeasibility. It may reduce the time execution when solving problems.

4.2 INTERFACE AND API FUNCTIONALITIES

The requirements to fulfil the functionalities in the previous section were achieved successfully as it will be explained in the following section.

Login and Register forms

The login form allows the user to enter the application while the register form enables new users to be registered.

Image 4: Login and register form.

Add facility

New facilities can be added, after selecting the “Add New Facility” button through the form seen in Image 5.

Image 5: Add New Facility form.

Map Interaction

The users facilities make an appearance on the map and they present a different icon depending on the type of facility they are: SF, RF, FF, A or FM (view Image 6).



Image 6: Types of facilities.

Each facility can be clicked and then a info-window is displayed presenting all the facility's characteristics.

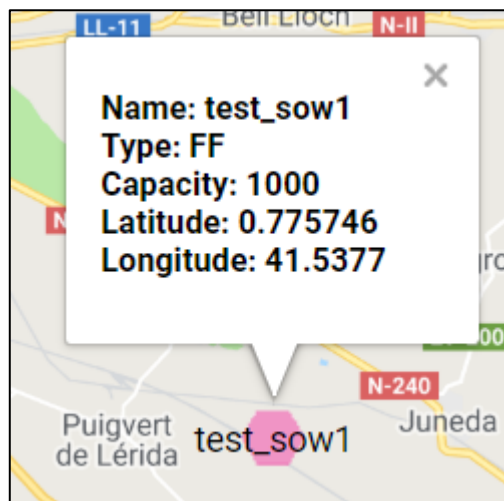


Image 7: Facility information.

In Image 8 the facility display can be seen, considering all kinds of facilities and the temporal marker, which acts as a guide to the user to add a new facility.



Image 8: Facilities and temporal marker display.

Get User Token

The user token is displayed when the “Show User Token” button is clicked. Then a window with the user token makes appearance. The token can be used to generate the data file through an Excel macro with the user’s facilities information required to run the model.

4.3 COMPUTATIONAL RESULTS

The instances were run again with the new winning configuration. In this section the execution time variation obtained of each instance will be analysed.

Instance	Sow Farms	Rearing Farms	Fattening Farms	Execution time old config (s)	Execution time new config (s)	Difference (s)	Variation (%)
1	2	6	12	4.75	0.99	-3.76	79%
2	2	6	20	9.53	1.35	-8.18	86%
3	2	8	20	13.61	2.16	-11.45	84%
4	2	10	40	14.95	8.09	-6.86	46%
5	2	15	80	29.81	23.59	-6.22	21%
6	3	8	15	31.53	2.70	-28.83	91%
7	3	6	10	39.02	60.58	21.56	-55%
8	3	15	20	44.94	4.72	-40.22	89%
9	4	12	20	54.81	5.33	-49.48	90%
10	3	15	40	71.25	18.82	-52.43	74%
11	3	15	90	82.17	39.92	-42.25	51%
12	3	15	55	96.81	27.41	-69.40	72%
13	3	15	60	107.47	35.84	-71.63	67%
14	3	18	80	136.94	57.08	-79.86	58%
15	4	12	40	159.17	12.23	-146.94	82%
16	4	15	60	173.19	37.92	-135.27	78%
17	3	21	82	183.19	66.47	-116.72	64%
18	4	15	40	200.14	19.42	-180.72	90%
19	4	8	15	203.28	83.68	-119.60	59%
20	3	21	85	216.55	77.16	-139.39	64%
21	4	18	85	228.39	68.47	-159.92	70%
22	3	21	80	233.61	59.49	-174.12	75%
23	4	12	80	247.23	31.89	-215.34	87%
24	4	18	60	256.36	52.57	-203.79	79%
25	5	18	50	301.17	37.24	-263.93	88%
26	5	18	60	304.38	45.62	-258.76	85%
27	5	21	70	473.20	67.56	-405.64	86%
28	5	18	70	496.28	58.53	-437.75	88%
29	5	18	90	564.28	77.69	-486.59	86%
30	5	18	80	628.24	64.51	-563.73	90%

Table 3: Execution times with the winning configuration and times variations.

In general terms, the instances execution time significantly drops. Actually, the average percental improvement in terms of execution time is 71.16%. However, an anomaly can be observed in the instance number 7 where the execution time increases instead of decrease. Therefore, even if in most cases the execution time improves, does not mean that in some isolated cases, instances will present worse execution times.

As we can see, just one out of the thirty instances execution time has decreased. This means that just 3.33% of the instances got worse while the remaining 96.66% have improved.

In addition, the fact that, apart from instance 7, the improvement interval goes from 51% to 90% is astonishing since it means that the execution time is reduced between a half and a tenth of the original execution time.

With these results so satisfactory, users will not have to remain senseless amounts of time waiting for the model to reach a solution.

4.4 WORKFLOW

The final distribution of the tasks and steps is the following one represented in a grant diagram, depending on the planned and the real time taken to develop the step or the task.

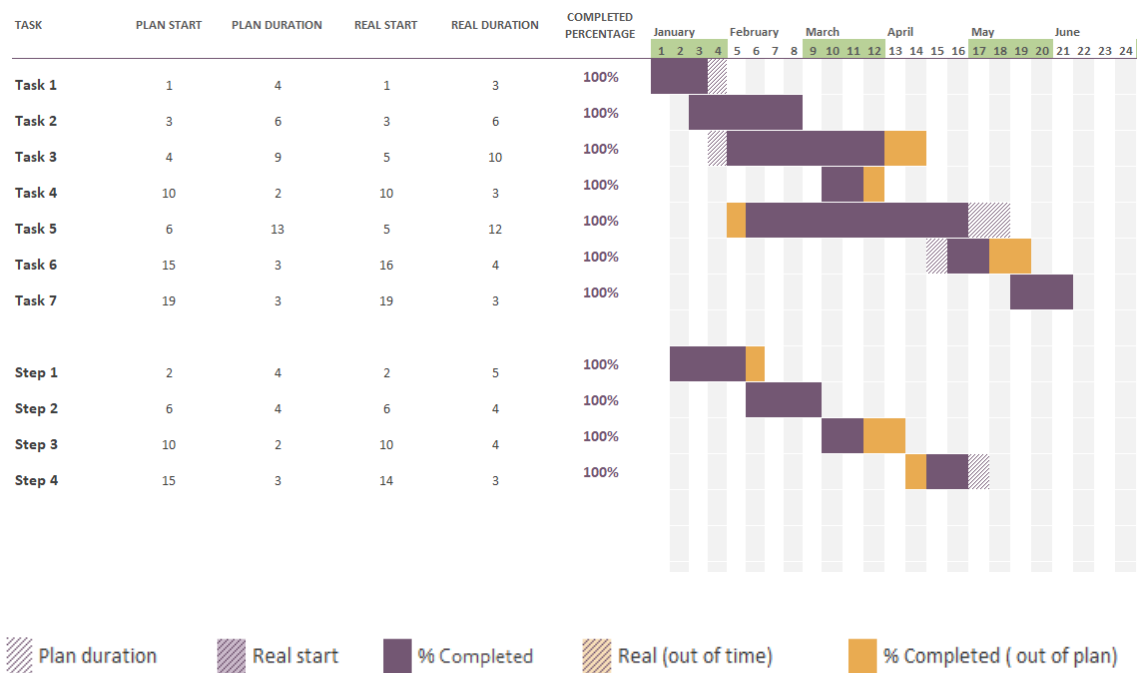


Image 9: Planned and Real tasks and steps time distribution.

5. CONCLUSIONS

As the project had two different parts of development due to the double degree requirements, the conclusion will also be divided in two sections.

In reference to the model part, the execution time has dropped significantly in every case, with just one exception. We are in the position of affirming that the configuration given, drastically reduces the execution time of the model to reasonable levels.

In major terms, applying the given configuration reduces the times of execution in average by a 71.3%. This reduction is successful in a 96.66% of the cases. Additionally, no case exceeds the minute and a half of execution time what means that in comparison with the default configuration, which exceeded the six hundred seconds in some cases (ten minutes), means that we are dealing with remarkable results.

With regard to the other part, the client application approaches with success the model to the users in a friendly and easy way. In addition, it provides an interface to manage their farms and facilities that pertain to their supply chain and visualize their location and characteristics in an interactive map.

Users are able to run the model with their own facilities information and get a personalized solution, which will be taken in consideration when making a decision.

In conclusion, both parts of the project have had positive and successful results. However, solutions should be flexible and adaptive to possible incoming changes. Therefore, the work does not end here as users necessities and requirements are continuously changing.

6. FUTURE WORK

Since the time of the project was limited, so have been the solutions, which they also changed over time. The project still has room for growing and improvement potential, mainly the part of the client application and the api.

For instance, the client app has the potential of processing the model results and show through the angular google maps module the routes of transport of the pigs. Currently, the information only goes one way, from the client to the api and then to the model. The implementation of this feature would lead to a two-way data flow as the client would generate the necessary data to run the model and then the model solution would generate the data to represent the solution in the client.

7. BIBLIOGRAPHY

- [1] Cécile Cornou and Anders Ringgaard Kristensen. Use of information from monitoring and decision support systems in pig production: Collection, applications and expected benefits. *Livestock Science*, 157(2-3):552–567, 2013
- [2] François Dubeau, Pierre Olivier Julien, and Candido Pomar. Formulating diets for growing pigs: Economic and environmental considerations. *Annals of Operations Research*, 190(1):239–269, 2011.
- [3] James Gray, Thomas M. Banhazi, and Alexander A. Kist. Wireless data management system for environmental monitoring in livestock buildings. *Information Processing in Agriculture*, 4(1):1–17, 2017.
- [4] Anders Ringgaard Kristensen and Thomas Algot Søllested. A sow replacement model using Bayesian updating in a three-level hierarchic Markov process: II. Optimization model. *Livestock Production Science*, 87(1):25–36, 2004.
- [5] P. Luatkep, A. Sumalee, W. H. Lam, Z.-C. Li, H. K. Lo, Global optimization method for mixed transportation network design problem: A mixed-integer linear programming approach, *Transportation Research Part B: Methodological* 45 (5) (2011) 808–827.
- [6] Hai-ning. KONG, A Green Mixed Integer Linear Programming Model for Optimization of Byproduct Gases in Iron and Steel Industry, *Journal of Iron and Steel Research*, International 22 (8) (2015) 681–685. doi:10.1016/S1006-706X(15)30057-1.
URL <http://linkinghub.elsevier.com/retrieve/pii/S1006706X15300571>
- [7] S. Fountas, G. Carli, C.G. Sørensen, Z. Tsiropoulos, C. Cavalaris, A. Vatsanidou, B. Liakos, M. Canavari, J. Wiebensohn, and B. Tisserye. Farm management information systems: Current situation and future perspectives. *Computers and Electronics in Agriculture*, 115:40–50, 2015.
- [8] Lluís M. Plà-Aragónés, editor. *Handbook of Operations Research in Agriculture and the Agri-Food Industry*, volume 224 of *International Series in Operations Research & Management Science*. Springer New York, New York, NY, 2015.
- [9] D. Plà, L.M. & Romero, Planning modern intensive livestock production: The case of the Spanish pig sector, *International Workshop in OR*. La Havana. Cuba
- [10] E. Nadal-Roig, L. M. Plà, Multiperiod planning tool for multisite pig production systems, *Journal of Animal Science* 92 (9) (2014) 4154–4160. doi:10.2527/jas.2014-7784.
- [11] B. Manos, K. Paparrizos, N. Matsatsinis, and J. Papathanasiou. *Decision support systems in agriculture, food and the environment: Trends, applications and advances*. 2010.

[12] J.A.L.M Kamp. Knowledge based systems: from research to practical application: pitfalls and critical success factors. *Computers and Electronics in Agriculture*, 22(2-3):243–250, 1999.

[13] James W. Jones, John M. Antle, Bruno Basso, Kenneth J. Boote, Richard T. Conant, Ian Foster, H. Charles J. Godfray, Mario Herrero, Richard E. Howitt, Sander Janssen, Brian A. Keating, Rafael Munoz-Carpena, Cheryl H. Porter, Cynthia Rosenzweig, and Tim R. Wheeler. Toward a new generation of agricultural system data, models, and knowledge products: State of agricultural systems science. *Agricultural Systems*, 155:269–288, 2017

Other references:

IBM Support - IBM Knowledge Center – List of CPLEX parameters. (n.d.) from https://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.3/ilog.odms.cplex.help/CPLEX/Parameters/topics/introListAlpha.html

ANNEX I: OPTIONS FILE

```

parameters.advance=0
parameters.barrier.algorithm=3
parameters.barrier.crossover=1
parameters.barrier.ordering=2
parameters.barrier.startalg=1
parameters.emphasis.mip=4
parameters.feasopt.mode=4
parameters.lpmethod=0
parameters.mip.cuts.cliques=1
parameters.mip.cuts.covers=-1
parameters.mip.cuts.disjunctive=2
parameters.mip.cuts.flowcovers=2
parameters.mip.cuts.gomory=2
parameters.mip.cuts.gubcovers=0
parameters.mip.cuts.implied=2
parameters.mip.cuts.liftproj=3
parameters.mip.cuts.localimplied=0
parameters.mip.cuts.mcfcut=1
parameters.mip.cuts.mircut=-1
parameters.mip.cuts.pathcut=1
parameters.mip.cuts.zerohalfcut=0
parameters.mip.limits.aggforcut=347
parameters.mip.limits.gomorycand=29
06
parameters.mip.limits.strongcand=40
parameters.mip.limits.submipnodelim=
2038
parameters.mip.ordertype=2
parameters.mip.pool.intensity=4
parameters.mip.strategy.bbinterval=94
4
parameters.mip.strategy.branch=0
parameters.mip.strategy.dive=3
parameters.mip.strategy.fphour=-1
parameters.mip.strategy.heuristicfreq=
3575
parameters.mip.strategy.lbhour=0
parameters.mip.strategy.nodeselect=1
parameters.mip.strategy.order=1
parameters.mip.strategy.presolvenode
=-1
parameters.mip.strategy.probe=-1
parameters.mip.strategy.rinshour=6969
parameters.mip.strategy.search=2
parameters.mip.strategy.startalgorithm
=4
parameters.mip.strategy.subalgorithm=
4
parameters.mip.strategy.variableselect
=1
parameters.preprocessing.aggregator=
6
parameters.preprocessing.boundstreng
th=1
parameters.preprocessing.coeffreduce
=2

```

```
parameters.preprocessing.dependency  
=0  
  
parameters.preprocessing.dual=0  
  
parameters.preprocessing.fill=785  
  
parameters.preprocessing.presolve=1  
  
parameters.preprocessing.reduce=1  
  
parameters.preprocessing.relax=1  
  
parameters.preprocessing.repeatpresol  
ve=-1  
  
parameters.preprocessing.symmetry=0  
  
parameters.read.scale=-1  
  
parameters.simplex.crash=1 (default)  
  
parameters.simplex.dgradient=4  
  
parameters.simplex.limits.singularity=1  
67  
  
parameters.simplex.perturbation.const  
ant=5  
  
parameters.simplex.perturbation.indica  
tor=0  
  
parameters.simplex.pgradient=-1  
  
parameters.simplex.refactor=409  
  
parameters.solutiontype=2
```